Chapitre 3. Mots et langages.

Lycée Fabert - MP* - MP

Novembre 2020

Ce chapitre traite des mots et des langages

Ce chapitre traite des mots et des langages

La question qui nous intéressera le plus est la suivante :

Ce chapitre traite des mots et des langages

La question qui nous intéressera le plus est la suivante : un mot donné appartient-il à un langage donné ?

Exemple 1 : Compilation

- But de la compilation

Exemple 1 : Compilation

- But de la compilation
- Première étape de la compilation :

Exemple 1 : Compilation

- But de la compilation
- Première étape de la compilation : analyse lexicale
- Deuxième étape de la compilation :

Exemple 1 : Compilation

- But de la compilation
- Première étape de la compilation : analyse lexicale
- Deuxième étape de la compilation : analyse syntaxique

Exemple 2 : Bio-informatique

Exemple 2 : Bio-informatique

- Composition d'un chromosome.

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN très longue séquence de lettres dans $\{G, A, C, T\}$

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN très longue séquence de lettres dans $\{G, A, C, T\}$
- Question 1 : comment déterminer la présence ou non d'un gène dans le patrimoine génétique d'un individu ?

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN très longue séquence de lettres dans $\{G, A, C, T\}$
- Question 1 : comment déterminer la présence ou non d'un gène dans le patrimoine génétique d'un individu ? (autrement dit rechercher une séquence particulière de nucléotides dans un chromosome)

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN très longue séquence de lettres dans $\{G, A, C, T\}$
- Question 1 : comment déterminer la présence ou non d'un gène dans le patrimoine génétique d'un individu ? (autrement dit rechercher une séquence particulière de nucléotides dans un chromosome)
- Question 2 : détecter des ressemblances entre plusieurs fragments d'ADN

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN très longue séquence de lettres dans $\{G, A, C, T\}$
- Question 1 : comment déterminer la présence ou non d'un gène dans le patrimoine génétique d'un individu ? (autrement dit rechercher une séquence particulière de nucléotides dans un chromosome)
- Question 2 : détecter des ressemblances entre plusieurs fragments d'ADN (pour localiser des gênes remplissant des mêmes fonctions ou faire des tests de familiarité)

Exemple 2 : Bio-informatique

- Composition d'un chromosome. Il existe quatre bases azotées différentes : la guanine G, l'adénine A, la cytosine C et la thymine T.
- Modélisation d'un brin d'ADN très longue séquence de lettres dans $\{G, A, C, T\}$
- Question 1 : comment déterminer la présence ou non d'un gène dans le patrimoine génétique d'un individu ? (autrement dit rechercher une séquence particulière de nucléotides dans un chromosome)
- Question 2 : détecter des ressemblances entre plusieurs fragments d'ADN (pour localiser des gênes remplissant des mêmes fonctions ou faire des tests de familiarité)

Exemple 3 : Recherche de motifs dans un texte

Exemple 3 : Recherche de motifs dans un texte

- Fonction rechercher des éditeurs de texte

Exemple 3 : Recherche de motifs dans un texte

- Fonction rechercher des éditeurs de texte
- Parfois permet de rechercher des ensembles de mots vérifiant des propriétés :

Exemple 3 : Recherche de motifs dans un texte

- Fonction rechercher des éditeurs de texte
- Parfois permet de rechercher des ensembles de mots vérifiant des propriétés : la notion d'expression régulière pour exprimer l'objet de la recherche

Exemple 3 : Recherche de motifs dans un texte

- Fonction rechercher des éditeurs de texte
- Parfois permet de rechercher des ensembles de mots vérifiant des propriétés : la notion d'expression régulière pour exprimer l'objet de la recherche
- Le motif de la recherche n'est pas forcément alphanumérique : (code-barre dans une image) .

Exemple 3 : Recherche de motifs dans un texte

- Fonction rechercher des éditeurs de texte
- Parfois permet de rechercher des ensembles de mots vérifiant des propriétés : la notion d'expression régulière pour exprimer l'objet de la recherche
- Le motif de la recherche n'est pas forcément alphanumérique : (code-barre dans une image) .Il peut aussi être mutidimensionnel (QR code)

Définition

- *Un alphabet Σ est un ensemble fini. Les éléments de Σ sont appelés lettres ou caractères.*

Définition

- *Un alphabet Σ est un ensemble fini. Les éléments de Σ sont appelés lettres ou caractères.*
- *Un mot sur un alphabet Σ est soit le mot vide noté ε (ou 1_Σ), soit une suite finie $u_1 u_2 \dots u_n$ de lettres*

Définition

- *Un alphabet Σ est un ensemble fini. Les éléments de Σ sont appelés lettres ou caractères.*
- *Un mot sur un alphabet Σ est soit le mot vide noté ε (ou 1_Σ), soit une suite finie $u_1 u_2 \dots u_n$ de lettres*
- *L'ensemble des mots non vides sur l'alphabet Σ est noté Σ^+ et on note $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.*

Définition

- *Un alphabet Σ est un ensemble fini. Les éléments de Σ sont appelés lettres ou caractères.*
- *Un mot sur un alphabet Σ est soit le mot vide noté ε (ou 1_Σ), soit une suite finie $u_1 u_2 \dots u_n$ de lettres*
- *L'ensemble des mots non vides sur l'alphabet Σ est noté Σ^+ et on note $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.*
- *Si u est un mot sur l'alphabet Σ, on appelle longueur de u, notée $|u|$ l'entier qui vaut 0 si $u = \varepsilon$ et qui vaut n si $u = u_1 u_2 \dots u_n$.*

Définition

- *Un alphabet Σ est un ensemble fini. Les éléments de Σ sont appelés lettres ou caractères.*
- *Un mot sur un alphabet Σ est soit le mot vide noté ε (ou 1_Σ), soit une suite finie $u_1 u_2 \dots u_n$ de lettres*
- *L'ensemble des mots non vides sur l'alphabet Σ est noté Σ^+ et on note $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.*
- *Si u est un mot sur l'alphabet Σ, on appelle longueur de u, notée $|u|$ l'entier qui vaut 0 si $u = \varepsilon$ et qui vaut n si $u = u_1 u_2 \dots u_n$.*
- *Si $n \in \mathbb{N}$, l'ensemble des mots de longueur n sur l'alphabet Σ est noté Σ^n.*

Définition

- *Un alphabet Σ est un ensemble fini. Les éléments de Σ sont appelés lettres ou caractères.*
- *Un mot sur un alphabet Σ est soit le mot vide noté ε (ou 1_Σ), soit une suite finie $u_1 u_2 \dots u_n$ de lettres*
- *L'ensemble des mots non vides sur l'alphabet Σ est noté Σ^+ et on note $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.*
- *Si u est un mot sur l'alphabet Σ, on appelle longueur de u, notée $|u|$ l'entier qui vaut 0 si $u = \varepsilon$ et qui vaut n si $u = u_1 u_2 \dots u_n$.*
- *Si $n \in \mathbb{N}$, l'ensemble des mots de longueur n sur l'alphabet Σ est noté Σ^n.*
- *Si u est un mot sur l'alphabet Σ et $a \in \Sigma$, on appelle longueur en a de u et on note $|u|_a$ l'entier égal à 0 si $u = \varepsilon$ et égal à $\text{card}\{k \in [\![1, n]\!] \mid u_k = a\}$ si $u = u_1 u_2 \dots u_n$*

Exemple : Sur l'alphabet $\Sigma = \{a, b, c, d\}$, $u = cabada$ est de longueur $|u| = 6$ et sa longueur en a est $|u|_a = 3$.

Exemple : Sur l'alphabet $\Sigma = \{a, b, c, d\}$, $u = cabada$ est de longueur $|u| = 6$ et sa longueur en a est $|u|_a = 3$.

Remarques : – on a $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \Sigma$, $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$ et $\Sigma^+ = \bigcup_{n \in \mathbb{N}^*} \Sigma^n$.

Exemple : Sur l'alphabet $\Sigma = \{a, b, c, d\}$, $u = cabada$ est de longueur $|u| = 6$ et sa longueur en a est $|u|_a = 3$.

Remarques : – on a $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \Sigma$, $\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$ et $\Sigma^+ = \bigcup_{n\in\mathbb{N}^*} \Sigma^n$.

– Pour tout mot u sur l'alphabet Σ, $|u| = \sum_{a\in\Sigma} |u|_a$.

Exemple : Sur l'alphabet $\Sigma = \{a, b, c, d\}$, $u = cabada$ est de longueur $|u| = 6$ et sa longueur en a est $|u|_a = 3$.

Remarques : – on a $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \Sigma$, $\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$ et $\Sigma^+ = \bigcup_{n\in\mathbb{N}^*} \Sigma^n$.

– Pour tout mot u sur l'alphabet Σ, $|u| = \sum_{a\in\Sigma} |u|_a$.

– On utilise souvent les lettres du début de l'alphabet courant $\{a, b, c, \ldots\}$ pour désigner des lettres et celles de la fin de l'alphabet pour désigner des mots $(u, v, w, \ldots)$.

Définition (Concaténation)

Si $u = u_1 \dots u_p$ et $v = v_1 \dots v_q$ sont deux mots sur l'alphabet Σ de longueurs p et q, on appelle concaténation de u et v le mot de longueur $n + p$ noté $u.v$ ou uv égal à $u.v = u_1 \dots u_p v_1 \dots v_q$.

Définition (Concaténation)

Si $u = u_1 \dots u_p$ et $v = v_1 \dots v_q$ sont deux mots sur l'alphabet Σ de longueurs p et q, on appelle concaténation de u et v le mot de longueur $n + p$ noté $u.v$ ou uv égal à $u.v = u_1 \dots u_p v_1 \dots v_q$.
Par ailleurs, pour tout mot $u \in \Sigma^*$, $u.\varepsilon = \varepsilon.u = u$.

Définition (Concaténation)

Si $u = u_1 \dots u_p$ et $v = v_1 \dots v_q$ sont deux mots sur l'alphabet Σ de longueurs p et q, on appelle concaténation de u et v le mot de longueur $n + p$ noté $u.v$ ou uv égal à $u.v = u_1 \dots u_p v_1 \dots v_q$.
Par ailleurs, pour tout mot $u \in \Sigma^*$, $u.\varepsilon = \varepsilon.u = u$.

Proposition

La concaténation est une loi de composition interne associative sur Σ^*, d'élément neutre ε.

Définition (Concaténation)

Si $u = u_1 \dots u_p$ et $v = v_1 \dots v_q$ sont deux mots sur l'alphabet Σ de longueurs p et q, on appelle concaténation de u et v le mot de longueur $n + p$ noté $u.v$ ou uv égal à $u.v = u_1 \dots u_p v_1 \dots v_q$.
Par ailleurs, pour tout mot $u \in \Sigma^*$, $u.\varepsilon = \varepsilon.u = u$.

Proposition

La concaténation est une loi de composition interne associative sur Σ^*, d'élément neutre ε.

Remarques : – Grâce à l'associativité de la concaténation, si $u \in \Sigma^*$, on peut définir par récurrence le mot u^n pour $n \in \mathbb{N}$ par : $u^0 = \varepsilon$ et pour tout $n \in \mathbb{N}$, $u^{n+1} = u^n.u$.
On a alors pour tout $(n, p) \in \mathbb{N}^2$, $u^{n+p} = u^n.u^p = u^p.u^n$.

Définition (Concaténation)

Si $u = u_1 \dots u_p$ et $v = v_1 \dots v_q$ sont deux mots sur l'alphabet Σ de longueurs p et q, on appelle concaténation de u et v le mot de longueur $n + p$ noté $u.v$ ou uv égal à $u.v = u_1 \dots u_p v_1 \dots v_q$.
Par ailleurs, pour tout mot $u \in \Sigma^*$, $u.\varepsilon = \varepsilon.u = u$.

Proposition
La concaténation est une loi de composition interne associative sur Σ^*, d'élément neutre ε.

Remarques : – Grâce à l'associativité de la concaténation, si $u \in \Sigma^*$, on peut définir par récurrence le mot u^n pour $n \in \mathbb{N}$ par : $u^0 = \varepsilon$ et pour tout $n \in \mathbb{N}$, $u^{n+1} = u^n.u$.
On a alors pour tout $(n, p) \in \mathbb{N}^2$, $u^{n+p} = u^n.u^p = u^p.u^n$.
– Seul le mot vide possède un symétrique pour la concaténation, mais tout mot est simplifiable à gauche ou à droite (c'est-à-dire que $uv = uw$ ou $vu = wu$ entraîne $v = w$)

Définition

Si u et v sont deux mots sur l'alphabet Σ, on dit que u est un

- *préfixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = uw$*

Définition

Si u et v sont deux mots sur l'alphabet Σ, on dit que u est un

- *préfixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = uw$*
- *suffixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = wu$*

Définition

Si u et v sont deux mots sur l'alphabet Σ, on dit que u est un

- *préfixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = uw$*
- *suffixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = wu$*
- *facteur de v s'il existe deux mots x et y de Σ^* tel que $v = xuy$*

Définition

Si u et v sont deux mots sur l'alphabet Σ, on dit que u est un

- *préfixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = uw$*
- *suffixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = wu$*
- *facteur de v s'il existe deux mots x et y de Σ^* tel que $v = xuy$*

Un préfixe (resp. suffixe) est dit propre si $w \neq \varepsilon$; un facteur est dit propre si $x \neq \varepsilon$ ou $y \neq \varepsilon$.

Définition

Si u et v sont deux mots sur l'alphabet Σ, on dit que u est un

- *préfixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = uw$*
- *suffixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = wu$*
- *facteur de v s'il existe deux mots x et y de Σ^* tel que $v = xuy$*

Un préfixe (resp. suffixe) est dit propre si $w \neq \varepsilon$; un facteur est dit propre si $x \neq \varepsilon$ ou $y \neq \varepsilon$.
Enfin, un sous-mot d'un mot $u = u_1 \dots u_n$ de longueur n est un mot $v = v_1 \dots v_p$ pour lequel il existe une application φ strictement croissante de $[\![1, p]\!]$ dans $[\![1, n]\!]$ telle que pour tout $k \in [\![1, p]\!]$, $v_k = u_{\varphi(k)}$.

Définition

Si u et v sont deux mots sur l'alphabet Σ, on dit que u est un

- *préfixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = uw$*
- *suffixe de v s'il existe un mot $w \in \Sigma^*$ tel que $v = wu$*
- *facteur de v s'il existe deux mots x et y de Σ^* tel que $v = xuy$*

Un préfixe (resp. suffixe) est dit propre si $w \neq \varepsilon$; un facteur est dit propre si $x \neq \varepsilon$ ou $y \neq \varepsilon$.
Enfin, un sous-mot d'un mot $u = u_1 \dots u_n$ de longueur n est un mot $v = v_1 \dots v_p$ pour lequel il existe une application φ strictement croissante de $[\![1, p]\!]$ dans $[\![1, n]\!]$ telle que pour tout $k \in [\![1, p]\!]$, $v_k = u_{\varphi(k)}$.

Exemples : "strophe" et "roi" sont des sous-mots de "claustrophobie" mais n'en sont pas des facteurs.

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.*

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Démonstration : $\diamond$ **Existence** Supposons par exemple que $|u| \geqslant |x|$.

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Démonstration : $\diamond$ **Existence** Supposons par exemple que $|u| \geqslant |x|$. Alors x, qui est un préfixe de uv, est un préfixe de u donc il existe $t \in \Sigma^*$ tel que $u = xt$.

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Démonstration : $\diamond$ **Existence** Supposons par exemple que $|u| \geqslant |x|$. Alors x, qui est un préfixe de uv, est un préfixe de u donc il existe $t \in \Sigma^*$ tel que $u = xt$.Mais alors, l'égalité $xy = uv$ s'écrit $xtv = xy$ donc, en simplifiant, $tv = y$.

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Démonstration : $\diamond$ **Existence** Supposons par exemple que $|u| \geqslant |x|$. Alors x, qui est un préfixe de uv, est un préfixe de u donc il existe $t \in \Sigma^*$ tel que $u = xt$.Mais alors, l'égalité $xy = uv$ s'écrit $xtv = xy$ donc, en simplifiant, $tv = y$.
Le cas où $|u| \leqslant |x|$ se traite de manière analogue.

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Démonstration : $\diamond$ **Existence** Supposons par exemple que $|u| \geqslant |x|$. Alors x, qui est un préfixe de uv, est un préfixe de u donc il existe $t \in \Sigma^*$ tel que $u = xt$.Mais alors, l'égalité $xy = uv$ s'écrit $xtv = xy$ donc, en simplifiant, $tv = y$.
Le cas où $|u| \leqslant |x|$ se traite de manière analogue.
$\diamond$ **Unicité**

Proposition

Lemme (*Levi*) *Soient x, y, u, v quatre mots sur Σ tels que $xy = uv$.Alors il existe un unique mot $t \in \Sigma^*$ tel que l'une des deux conditions suivantes soit réalisée :*

- $u = xt$ *et* $y = tv$
- $x = ut$ *et* $v = ty$

Shéma illustrant cette propriété :

Démonstration : $\diamond$ **Existence** Supposons par exemple que $|u| \geqslant |x|$. Alors x, qui est un préfixe de uv, est un préfixe de u donc il existe $t \in \Sigma^*$ tel que $u = xt$.Mais alors, l'égalité $xy = uv$ s'écrit $xtv = xy$ donc, en simplifiant, $tv = y$.
Le cas où $|u| \leqslant |x|$ se traite de manière analogue.
$\diamond$ **Unicité**

Corollaire Si x et y sont deux préfixes d'un mot u, alors x est préfixe de y ou y est préfixe de x.

Théorème

Soient $x, y, z \in \Sigma^*$ *tels que* $xy = yz$ *et* $x \neq \varepsilon$.

Théorème

Soient $x, y, z \in \Sigma^*$ *tels que* $xy = yz$ *et* $x \neq \varepsilon$. *Alors* $\exists u, v \in \Sigma^*$ *et* $k \in \mathbb{N}$ *tels que :* $x = uv$, $y = (uv)^k u = u(vu)^k$ *et* $z = vu$

Théorème

Soient $x, y, z \in \Sigma^*$ tels que $xy = yz$ et $x \neq \varepsilon$. Alors $\exists u, v \in \Sigma^*$ et $k \in \mathbb{N}$ tels que : $x = uv$, $y = (uv)^k u = u(vu)^k$ et $z = vu$

Démonstration : $\diamond$ Si $|x| \geqslant |y|$, on peut appliquer le théorème de Levi : il existe $t \in \Sigma^*$ tel que $x = yt$ et $z = ty$ d'où le résultat souhaité avec $u = y$, $v = t$ et $k = 0$.

Théorème

Soient $x, y, z \in \Sigma^*$ tels que $xy = yz$ et $x \neq \varepsilon$. Alors $\exists u, v \in \Sigma^*$ et $k \in \mathbb{N}$ tels que : $x = uv$, $y = (uv)^k u = u(vu)^k$ et $z = vu$

Démonstration : $\diamond$ Si $|x| \geqslant |y|$, on peut appliquer le théorème de Levi : il existe $t \in \Sigma^*$ tel que $x = yt$ et $z = ty$ d'où le résultat souhaité avec $u = y$, $v = t$ et $k = 0$.
$\diamond$ Si $|x| \leqslant |y|$ on montre le résultat par récurrence sur $|y|$.

Théorème

Soient $x, y, z \in \Sigma^*$ tels que $xy = yz$ et $x \neq \varepsilon$. Alors $\exists u, v \in \Sigma^*$ et $k \in \mathbb{N}$ tels que : $x = uv$, $y = (uv)^k u = u(vu)^k$ et $z = vu$

Démonstration : $\diamond$ Si $|x| \geqslant |y|$, on peut appliquer le théorème de Levi : il existe $t \in \Sigma^*$ tel que $x = yt$ et $z = ty$ d'où le résultat souhaité avec $u = y$, $v = t$ et $k = 0$.

$\diamond$ Si $|x| \leqslant |y|$ on montre le résultat par récurrence sur $|y|$.

- Si $|y| = 1$, on a nécessairement $|x| = 1$ (car $x \neq \varepsilon$) et alors $x = y = z$. Le résultat est acquis avec $u = y$, $v = \varepsilon$ et $k = 0$.

Théorème

Soient $x, y, z \in \Sigma^*$ tels que $xy = yz$ et $x \neq \varepsilon$. Alors $\exists u, v \in \Sigma^*$ et $k \in \mathbb{N}$ tels que : $x = uv$, $y = (uv)^k u = u(vu)^k$ et $z = vu$

Démonstration : $\diamond$ Si $|x| \geqslant |y|$, on peut appliquer le théorème de Levi : il existe $t \in \Sigma^*$ tel que $x = yt$ et $z = ty$ d'où le résultat souhaité avec $u = y$, $v = t$ et $k = 0$.

$\diamond$ Si $|x| \leqslant |y|$ on montre le résultat par récurrence sur $|y|$.

- Si $|y| = 1$, on a nécessairement $|x| = 1$ (car $x \neq \varepsilon$) et alors $x = y = z$. Le résultat est acquis avec $u = y$, $v = \varepsilon$ et $k = 0$.
- Soit $j \in \mathbb{N}^*$ tel que le résultat soit vrai pour tout mot y de longueur $\leqslant j$ et soit y un mot de longueur égale à $j + 1$. D'après le lemme de Levi, il existe $t \in \Sigma^*$ tel que $y = xt$ et $y = tz$. On a donc $xt = tz$ et puisque $x \neq \varepsilon$, $|t| \leqslant j$. D'après l'hypothèse de récurrence, il existe deux mots u et v et $k \in \mathbb{N}$ tels que $x = uv$, $t = (uv)^k u = u(vu)^k$ et $z = vu$. Il reste à calculer $y = xt = tz$ pour obtenir $y = (uv)^{k+1} u = u(vu)^{k+1}$. ♯

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$.

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

Démonstration : Par récurrence sur $|xy|$

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

Démonstration : Par récurrence sur $|xy|$

$\diamond$ Si $|xy| = 2$, alors $|x| = |y| = 1$ et $x = y$. Le résultat est acquis en posant $u = x = y$ et $i = j = 1$.

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

Démonstration : Par récurrence sur $|xy|$

$\diamond$ Si $|xy| = 2$, alors $|x| = |y| = 1$ et $x = y$. Le résultat est acquis en posant $u = x = y$ et $i = j = 1$.

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

$\diamond$ Soit $m \geqslant 2$ tel que la propriété soit vraie pour tout couple (x, y) de mots tels que $|xy| \leqslant m$. Soit $x, y \in \Sigma^+$ tels que $xy = yx$ et $|xy| = m + 1$:

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

$\diamond$ Soit $m \geqslant 2$ tel que la propriété soit vraie pour tout couple (x, y) de mots tels que $|xy| \leqslant m$. Soit $x, y \in \Sigma^+$ tels que $xy = yx$ et $|xy| = m + 1$:en appliquant le théorème précédent, il existe deux mots v, w et $k \in \mathbb{N}$ tels que $x = vw = wv$ et $y = (vw)^k v = v(wv)^k$.

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

$\diamond$ Soit $m \geqslant 2$ tel que la propriété soit vraie pour tout couple (x, y) de mots tels que $|xy| \leqslant m$. Soit $x, y \in \Sigma^+$ tels que $xy = yx$ et $|xy| = m + 1$:en appliquant le théorème précédent, il existe deux mots v, w et $k \in \mathbb{N}$ tels que $x = vw = wv$ et $y = (vw)^k v = v(wv)^k$.

- Si $v = \varepsilon$ alors $y = x^k$ et on a le résultat souhaité avec $u = x$, $i = 1$ et $j = k$.

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

$\diamond$ Soit $m \geqslant 2$ tel que la propriété soit vraie pour tout couple (x, y) de mots tels que $|xy| \leqslant m$. Soit $x, y \in \Sigma^+$ tels que $xy = yx$ et $|xy| = m + 1$:en appliquant le théorème précédent, il existe deux mots v, w et $k \in \mathbb{N}$ tels que $x = vw = wv$ et $y = (vw)^k v = v(wv)^k$.

- Si $v = \varepsilon$ alors $y = x^k$ et on a le résultat souhaité avec $u = x$, $i = 1$ et $j = k$.
- Si $w = \varepsilon$ alors $y = x^{k+1}$ et on a le résultat souhaité avec $u = x$, $i = 1$ et $j = k + 1$

Théorème

Soient x et y deux mots sur Σ tels que $xy = yx$, $x \neq \varepsilon$ et $y \neq \varepsilon$. Alors il existe un mot $u \in \Sigma^*$ et $(i, j) \in \mathbb{N}^2$ tels que $x = u^i$ et $y = u^j$.

$\diamond$ Soit $m \geqslant 2$ tel que la propriété soit vraie pour tout couple (x, y) de mots tels que $|xy| \leqslant m$. Soit $x, y \in \Sigma^+$ tels que $xy = yx$ et $|xy| = m + 1$:en appliquant le théorème précédent, il existe deux mots v, w et $k \in \mathbb{N}$ tels que $x = vw = wv$ et $y = (vw)^k v = v(wv)^k$.

- Si $v = \varepsilon$ alors $y = x^k$ et on a le résultat souhaité avec $u = x$, $i = 1$ et $j = k$.
- Si $w = \varepsilon$ alors $y = x^{k+1}$ et on a le résultat souhaité avec $u = x$, $i = 1$ et $j = k + 1$
- Sinon, $|vw| = |x| < |xy|$ (car $y \neq \varepsilon$) donc on peut appliquer l'hypothèse de récurrence : il existe un mot u et i et j tels que $v = u^i$ et $w = u^j$ et alors $x = u^{i+j}$ et $y = u^{(k+1)i+kj}$

But : On veut écrire une fonction déterminant le nombre d'occurences d'un motif donné dans un texte.

But : On veut écrire une fonction déterminant le nombre d'occurences d'un motif donné dans un texte.

Première idée : déplacer une fenêtre de longueur égale à celle du motif cherché.

let recherche motif texte =

But : On veut écrire une fonction déterminant le nombre d'occurences d'un motif donné dans un texte.

Première idée : déplacer une fenêtre de longueur égale à celle du motif cherché.

```
let recherche motif texte =
    let resultat = ref [] in
```

But : On veut écrire une fonction déterminant le nombre d'occurences d'un motif donné dans un texte.

Première idée : déplacer une fenêtre de longueur égale à celle du motif cherché.

```
let recherche motif texte =
    let resultat = ref [] in
    let n = String.length motif and p = String.length texte in
```

But : On veut écrire une fonction déterminant le nombre d'occurences d'un motif donné dans un texte.

Première idée : déplacer une fenêtre de longueur égale à celle du motif cherché.

```
let recherche motif texte =
  let resultat = ref [] in
  let n = String.length motif and p = String.length texte in
  for i = 0 to p - n do
    if String.sub texte i n = mot
    then resultat := i : :( !resultat)
```

But : On veut écrire une fonction déterminant le nombre d'occurences d'un motif donné dans un texte.

Première idée : déplacer une fenêtre de longueur égale à celle du motif cherché.

```
let recherche motif texte =
   let resultat = ref [] in
   let n = String.length motif and p = String.length texte in
   for i = 0 to p - n do
      if String.sub texte i n = mot
      then resultat := i : :( !resultat)
   done ;
 !resultat ; ;
```

Exemple :

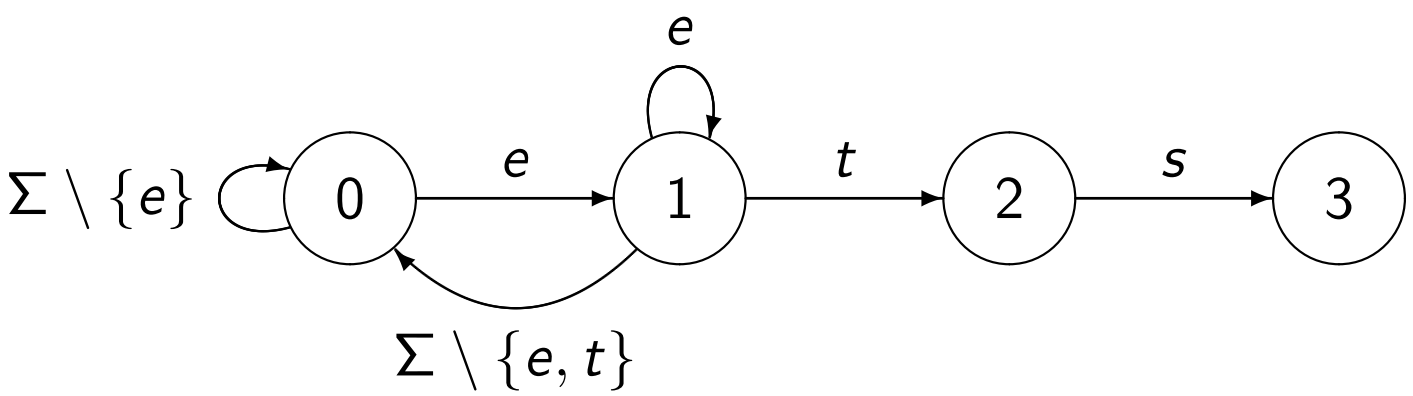
```
let m = "ets" ; ;
val m : string = "ets"
let t = "il met son pull car il ets frileux et ne veut pas avoir froid
pendant les derniers sets" ; ;
val t : string = "il met son pull car ... sets"
```

Exemple :

```
let m = "ets" ; ;
val m : string = "ets"
let t = "il met son pull car il ets frileux et ne veut pas avoir froid
pendant les derniers sets" ; ;
val t : string = "il met son pull car ... sets"
recherche m t ; ;
- : int list = [84 ; 23]
pause
```

Deuxième méthode : lire le texte lettre par lettre et dans laquelle on garde en mémoire le nombre de lettres du motif auquel on est parvenu.

Deuxième méthode : lire le texte lettre par lettre et dans laquelle on garde en mémoire le nombre de lettres du motif auquel on est parvenu.Représentation à l'aide d'un automate :



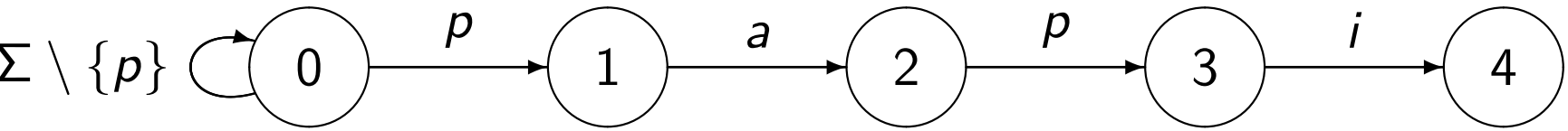
```
let recherche motif texte =
    let resultat = ref [] in
    let n = String.length motif and p = String.length texte in
    for i = 0 to p - n do
```

```
let recherche motif texte =
  let resultat = ref [] in
  let n = String.length motif and p = String.length texte in
  for i = 0 to p - n do
    let j = ref 0 in
    while !j < n && motif.[ !j] = t.[i + !j] do
      incr j
    done ;
```

```
let recherche motif texte =
  let resultat = ref [] in
  let n = String.length motif and p = String.length texte in
  for i = 0 to p - n do
    let j = ref 0 in
    while !j < n && motif.[ !j] = t.[i + !j] do
      incr j
    done ;
    if !j = n then resultat := i : :( !resultat)
  done ;
!resultat ; ;
```

Remarque : L'automate représentant la lecture d'un motif peut être plus compliqué. Compléter par exemple celui correspondant à la recherche du motif "papi" dans un texte.

Remarque : L'automate représentant la lecture d'un motif peut être plus compliqué. Compléter par exemple celui correspondant à la recherche du motif "papi" dans un texte.



Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit comme

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit commel'ensemble des préfixes non vides du mot $m = abab$.

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit commel'ensemble des préfixes non vides du mot $m = abab$.
- $L_2 = \{a^n b^n;\ n \in \mathbb{N}\}$ peut être décrit

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit commel'ensemble des préfixes non vides du mot $m = abab$.
- $L_2 = \{a^n b^n;\ n \in \mathbb{N}\}$ peut être décrit comme l'ensemble des mots commençant par un certain nombre de a suivis du même nombre de b.

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit commel'ensemble des préfixes non vides du mot $m = abab$.
- $L_2 = \{a^n b^n;\ n \in \mathbb{N}\}$ peut être décrit comme l'ensemble des mots commençant par un certain nombre de a suivis du même nombre de b.
- $L_3 = \{mba;\ m \in \Sigma^*\}$ est l'ensemble des mots

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit commel'ensemble des préfixes non vides du mot $m = abab$.
- $L_2 = \{a^n b^n;\ n \in \mathbb{N}\}$ peut être décrit comme l'ensemble des mots commençant par un certain nombre de a suivis du même nombre de b.
- $L_3 = \{mba;\ m \in \Sigma^*\}$ est l'ensemble des mots se terminant par ab.

Définition

On appelle langage sur un alphabet Σ toute partie de Σ^*.

Exemples : Un langage peut être écrit en énumérant ses éléments ou à l'aide d'une propriété caractérisant ses éléments. Par exemple sur l'alphabet $\Sigma = \{a, b\}$,

- $L_1 = \{a, ab, aba, abab\}$ peut également être décrit commel'ensemble des préfixes non vides du mot $m = abab$.
- $L_2 = \{a^n b^n;\ n \in \mathbb{N}\}$ peut être décrit comme l'ensemble des mots commençant par un certain nombre de a suivis du même nombre de b.
- $L_3 = \{mba;\ m \in \Sigma^*\}$ est l'ensemble des mots se terminant par ab.

On peut également décrire un langage par des règles de construction, qu'on appelle dans ce cas une grammaire formelle.

On peut également décrire un langage par des règles de construction, qu'on appelle dans ce cas une grammaire formelle.
Par exemple le langage L sur $\Sigma = \{a, b\}$ défini par :

$$\varepsilon \in L \text{ et } (r, s) \in L^2 \Rightarrow arbs \in L$$

est appelé langage des mots de DYCK.
On peut montrer par exemple que tout mot de DYCK est de longueur paire par récurrence sur la longueur n de ce mot.

On peut également décrire un langage par des règles de construction, qu'on appelle dans ce cas une grammaire formelle.
Par exemple le langage L sur $\Sigma = \{a, b\}$ défini par :

$$\varepsilon \in L \text{ et } (r, s) \in L^2 \Rightarrow arbs \in L$$

est appelé langage des mots de DYCK.
On peut montrer par exemple que tout mot de DYCK est de longueur paire par récurrence sur la longueur n de ce mot.
L'ensemble des mots de DYCK de longueur inférieure ou égale à 6 est :

On peut également décrire un langage par des règles de construction, qu'on appelle dans ce cas une grammaire formelle.
Par exemple le langage L sur $\Sigma = \{a, b\}$ défini par :

$$\varepsilon \in L \text{ et } (r, s) \in L^2 \Rightarrow arbs \in L$$

est appelé langage des mots de DYCK.
On peut montrer par exemple que tout mot de DYCK est de longueur paire par récurrence sur la longueur n de ce mot.
L'ensemble des mots de DYCK de longueur inférieure ou égale à 6 est :$\{\varepsilon, ab, abab, a^2b^2, a^2b^2ab, (ab)^3, aba^2b^2, a^2bab^2, a^3b^3\}$

Opérations sur les langages

Opérations ensemblistes

Opérations sur les langages

Opérations ensemblistes

Définition

Soit L et L' deux langages sur l'alphabet Σ

- *La concaténation de L et L' est le langage, noté LL' défini par*

$$LL' = \{uv,\ u \in L \text{ et } v \in L'\}$$

Opérations sur les langages

Opérations ensemblistes

Définition

Soit L et L' deux langages sur l'alphabet Σ

- *La concaténation de L et L' est le langage, noté LL' défini par*

$$LL' = \{uv,\ u \in L \text{ et } v \in L'\}$$

- *On définit, par récurrence, la suite $(L^n)_{n\in\mathbb{N}}$ de langages par*
  - $L^0 = \{\varepsilon\}$
  - $\forall\, n \in \mathbb{N},\ L^{n+1} = L^n L = L^n L$

Opérations sur les langages

Opérations ensemblistes

Définition

Soit L et L' deux langages sur l'alphabet Σ

- *La concaténation de L et L' est le langage, noté LL' défini par*

$$LL' = \{uv,\ u \in L \text{ et } v \in L'\}$$

- *On définit, par récurrence, la suite $(L^n)_{n\in\mathbb{N}}$ de langages par*
  - $L^0 = \{\varepsilon\}$
  - $\forall\, n \in \mathbb{N},\ L^{n+1} = L^n L = L^n L$
- *L'étoile (ou fermeture de Kleene) du langage L est le langage L^* défini par $L^* = \bigcup_{n\in\mathbb{N}} L^n$ et on note $L^+ = \bigcup_{n\in\mathbb{N}^*} L^n$.*

Remarques :

- Attention à ne pas confondre L^2 avec $\{u^2,\ u \in L\}$.

Remarques :

- Attention à ne pas confondre L^2 avec $\{u^2,\ u \in L\}$.Par exemple si $L = \{ab, ba\}$, $L^2 = \{abab, baba, abba, baab\}$ tandis que $\{u^2,\ u \in L\} = \{abab, baba\}$. Même remarque pour L^n.

Remarques :

- Attention à ne pas confondre L^2 avec $\{u^2,\ u \in L\}$.Par exemple si $L = \{ab, ba\}$, $L^2 = \{abab, baba, abba, baab\}$ tandis que $\{u^2,\ u \in L\} = \{abab, baba\}$. Même remarque pour L^n.
- L^* est l'ensemble des mots obtenus en concaténant un nombre fini (éventuellement réduit à 0) de mots de L.

Remarques :

- Attention à ne pas confondre L^2 avec $\{u^2,\ u \in L\}$.Par exemple si $L = \{ab, ba\}$, $L^2 = \{abab, baba, abba, baab\}$ tandis que $\{u^2,\ u \in L\} = \{abab, baba\}$. Même remarque pour L^n.
- L^* est l'ensemble des mots obtenus en concaténant un nombre fini (éventuellement réduit à 0) de mots de L.
- on a $L^+ = LL^*$.

Remarques :

- Attention à ne pas confondre L^2 avec $\{u^2,\ u \in L\}$.Par exemple si $L = \{ab, ba\}$, $L^2 = \{abab, baba, abba, baab\}$ tandis que $\{u^2,\ u \in L\} = \{abab, baba\}$. Même remarque pour L^n.
- L^* est l'ensemble des mots obtenus en concaténant un nombre fini (éventuellement réduit à 0) de mots de L.
- on a $L^+ = LL^*$.
- Contrairement à L^* qui contient toujours le mot vide, L^+ ne contient le mot vide que si L le contient.

Donnons encore deux opérations sur les langages.

Donnons encore deux opérations sur les langages.

Définition

Soit K, L deux langages sur un alphabet Σ.

- *On appelle racine carrée du langage L, notée $\sqrt{L}$ le langage défini par*

$$\sqrt{L} = \{u \in \Sigma^* \mid u^2 \in L\}$$

Donnons encore deux opérations sur les langages.

Définition

Soit K, L deux langages sur un alphabet Σ.

- *On appelle racine carrée du langage L, notée $\sqrt{L}$ le langage défini par*
$$\sqrt{L} = \{u \in \Sigma^* \mid u^2 \in L\}$$

- *On appelle quotient (gauche) des langages K et L, noté $K^{-1}L$ le langage*
$$K^{-1}L = \{v \in \Sigma^* \mid \exists u \in K \text{ tel que } uv \in L\}$$

Donnons encore deux opérations sur les langages.

Définition

Soit K, L deux langages sur un alphabet Σ.

- *On appelle racine carrée du langage L, notée $\sqrt{L}$ le langage défini par*
$$\sqrt{L} = \{u \in \Sigma^* \mid u^2 \in L\}$$

- *On appelle quotient (gauche) des langages K et L, noté $K^{-1}L$ le langage*
$$K^{-1}L = \{v \in \Sigma^* \mid \exists u \in K \text{ tel que } uv \in L\}$$

Exemple : Si $L = \{a^n b^n \; n \in \mathbb{N}\}$ et $K = \{am, \; m \in \Sigma^*\}$, déterminer $\sqrt{L}$ et $K^{-1}L$

Définition

Soit Σ un alphabet. Les expressions rationnelles (ou regulières) sur Σ sont définies inductivement par les propriétés suivantes :

Définition

Soit Σ un alphabet. Les expressions rationnelles (ou regulières) sur Σ sont définies inductivement par les propriétés suivantes :

- *$\emptyset$ et ε sont des expressions rationnelles,*

Définition

Soit Σ un alphabet. Les expressions rationnelles (ou regulières) sur Σ sont définies inductivement par les propriétés suivantes :

- *$\emptyset$ et ε sont des expressions rationnelles,*
- *$\forall\, a \in \Sigma$, a est une expression rationnelle,*

Définition

Soit Σ un alphabet. Les expressions rationnelles (ou regulières) sur Σ sont définies inductivement par les propriétés suivantes :

- *$\emptyset$ et ε sont des expressions rationnelles,*
- *$\forall\, a \in \Sigma$, a est une expression rationnelle,*
- *Si e_1 et e_2 sont des expressions rationnelles, alors $e_1 + e_2$, $e_1 e_2$ et e_1^* sont des expressions rationnelles.*

Définition

Soit Σ un alphabet. Les expressions rationnelles (ou regulières) sur Σ sont définies inductivement par les propriétés suivantes :

- *$\emptyset$ et ε sont des expressions rationnelles,*
- *$\forall\, a \in \Sigma$, a est une expression rationnelle,*
- *Si e_1 et e_2 sont des expressions rationnelles, alors $e_1 + e_2$, $e_1 e_2$ et e_1^* sont des expressions rationnelles.*

Remarque : $e_1 + e_2$ est parfois notée $e_1|e_2$ et e^* est également notée $e*$.

Définition

Soit Σ un alphabet. Les expressions rationnelles (ou regulières) sur Σ sont définies inductivement par les propriétés suivantes :

- *$\emptyset$ et ε sont des expressions rationnelles,*
- *$\forall\, a \in \Sigma$, a est une expression rationnelle,*
- *Si e_1 et e_2 sont des expressions rationnelles, alors $e_1 + e_2$, $e_1 e_2$ et e_1^* sont des expressions rationnelles.*

Remarque : $e_1 + e_2$ est parfois notée $e_1|e_2$ et e^* est également notée $e*$.

Expliquons maintenant comment on associe à toute expression rationnelle un langage.

Définition

L'interprétation d'une expression rationnelle est définie par les règles inductives suivantes :

- $\emptyset$ *dénote le langage vide et* ε *dénote le langage* $\{\varepsilon\}$,

Définition

L'interprétation d'une expression rationnelle est définie par les règles inductives suivantes :

- $\emptyset$ *dénote le langage vide et* ε *dénote le langage* $\{\varepsilon\}$,
- $\forall\, a \in \Sigma$, a *dénote le langage* $\{a\}$,

Définition

L'interprétation d'une expression rationnelle est définie par les règles inductives suivantes :

- $\emptyset$ *dénote le langage vide et* ε *dénote le langage* $\{\varepsilon\}$,
- $\forall\, a \in \Sigma$, a *dénote le langage* $\{a\}$,
- $e_1 + e_2$ *dénote la réunion des langages dénotés par* e_1 *et* e_2,

Définition

L'interprétation d'une expression rationnelle est définie par les règles inductives suivantes :

- $\emptyset$ *dénote le langage vide et* ε *dénote le langage* $\{\varepsilon\}$,
- $\forall\, a \in \Sigma$, a *dénote le langage* $\{a\}$,
- $e_1 + e_2$ *dénote la réunion des langages dénotés par* e_1 *et* e_2,
- $e_1 e_2$ *dénote la concaténation des langages dénotés par* e_1 *et* e_2

Définition

L'interprétation d'une expression rationnelle est définie par les règles inductives suivantes :

- $\emptyset$ *dénote le langage vide et* ε *dénote le langage* $\{\varepsilon\}$,
- $\forall\, a \in \Sigma$, a *dénote le langage* $\{a\}$,
- $e_1 + e_2$ *dénote la réunion des langages dénotés par* e_1 *et* e_2,
- $e_1 e_2$ *dénote la concaténation des langages dénotés par* e_1 *et* e_2
- e_1^* *dénote l'étoile de* KLEENE *du langage dénoté par* e_1.

Définition

L'interprétation d'une expression rationnelle est définie par les règles inductives suivantes :

- $\emptyset$ *dénote le langage vide et* ε *dénote le langage* $\{\varepsilon\}$,
- $\forall\, a \in \Sigma$, a *dénote le langage* $\{a\}$,
- $e_1 + e_2$ *dénote la réunion des langages dénotés par* e_1 *et* e_2,
- $e_1 e_2$ *dénote la concaténation des langages dénotés par* e_1 *et* e_2
- e_1^* *dénote l'étoile de* KLEENE *du langage dénoté par* e_1.

Si e *est une expression rationnelle sur l'alphabet* Σ, *on note* $L[e]$ *le langage dénoté par* e.

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote le langage des mots dont aa est un facteur

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote le langage des mots dont aa est un facteur
- $(a+ba)^*$ dénote

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote le langage des mots dont aa est un facteur
- $(a+ba)^*$ dénote le langage des mots dans lesquels chaque b et suivi obligatoirement d'un a.

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote le langage des mots dont aa est un facteur
- $(a+ba)^*$ dénote le langage des mots dans lesquels chaque b et suivi obligatoirement d'un a.
- $(a^*b)^*$ dénote

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote le langage des mots dont aa est un facteur
- $(a+ba)^*$ dénote le langage des mots dans lesquels chaque b et suivi obligatoirement d'un a.
- $(a^*b)^*$ dénote le langage formé du mot vide et des mots se terminant par b. Notons que $\varepsilon + (a+b)^*b$ dénote le même langage.

Exemples : Sur l'alphabet $\Sigma = \{a, b\}$,

- a^*b^* dénote le langage des mots commençant par un certain nombre (éventuellement nul) de a suivi d'un certain nombre (éventuellement nul) de b.
- $(ab)^*$ dénote le langage constitué du mot vide et des mots commençant par a, alternant les a et les b et terminant par b. Notons que $\varepsilon + a(ba)^*b$ dénote le même langage.
- $(a+b)^*aa(a+b)^*$ dénote le langage des mots dont aa est un facteur
- $(a+ba)^*$ dénote le langage des mots dans lesquels chaque b et suivi obligatoirement d'un a.
- $(a^*b)^*$ dénote le langage formé du mot vide et des mots se terminant par b. Notons que $\varepsilon + (a+b)^*b$ dénote le même langage.

Définition

Un langage est dit rationnel s'il est dénoté par une expression rationnelle.
L'ensemble des langages rationnels sur l'alphabet Σ est noté $Rat(\Sigma)$

Définition

Un langage est dit rationnel s'il est dénoté par une expression rationnelle.
L'ensemble des langages rationnels sur l'alphabet Σ est noté $Rat(\Sigma)$

Exemples :

- Σ est rationnel car il dénoté par l'expression rationnelle $\sum_{a \in \Sigma} a$.

Définition

Un langage est dit rationnel s'il est dénoté par une expression rationnelle.
L'ensemble des langages rationnels sur l'alphabet Σ est noté $Rat(\Sigma)$

Exemples :

- Σ est rationnel car il dénoté par l'expression rationnelle $\sum_{a \in \Sigma} a$.
- Plus généralement, tout langage fini est rationnel.

Définition

Un langage est dit rationnel s'il est dénoté par une expression rationnelle.
L'ensemble des langages rationnels sur l'alphabet Σ est noté $Rat(\Sigma)$

Exemples :

- Σ est rationnel car il dénoté par l'expression rationnelle $\sum_{a \in \Sigma} a$.
- Plus généralement, tout langage fini est rationnel.
- Comme Σ est rationnel, Σ^* est rationnel ainsi que $\Sigma^+ = \Sigma\Sigma^*$.

Définition

Un langage est dit rationnel s'il est dénoté par une expression rationnelle.
L'ensemble des langages rationnels sur l'alphabet Σ est noté $Rat(\Sigma)$

Exemples :

- Σ est rationnel car il dénoté par l'expression rationnelle $\sum_{a \in \Sigma} a$.
- Plus généralement, tout langage fini est rationnel.
- Comme Σ est rationnel, Σ^* est rationnel ainsi que $\Sigma^+ = \Sigma\Sigma^*$.
- L'ensemble des mots de longueur impaire est rationnel car il est égal à $\Sigma(\Sigma^2)^*$

Théorème

L'ensemble des langages rationnels sur Σ est la plus petite partie de $\mathscr{P}(\Sigma^*)$ (au sens de l'inclusion) contenant $\emptyset$, $\{\varepsilon\}$, $\{a\}$ pour tout $a \in \Sigma$ et qui est stable par réunion, concaténation et passage à l'étoile de KLEENE.

Arbre binaire associé à une expression rationnelle

On peut naturellement associer à toute expression rationnelle un arbre binaire où les feuilles sont éléments de $\{\emptyset, \varepsilon\} \cup \Sigma$ et les nœuds les opérateurs $\{+, ., *\}$.

Arbre binaire associé à une expression rationnelle

On peut naturellement associer à toute expression rationnelle un arbre binaire où les feuilles sont éléments de $\{\emptyset, \varepsilon\} \cup \Sigma$ et les nœuds les opérateurs $\{+, ., *\}$.

Intérêt .

Arbre binaire associé à une expression rationnelle

On peut naturellement associer à toute expression rationnelle un arbre binaire où les feuilles sont éléments de $\{\emptyset, \varepsilon\} \cup \Sigma$ et les nœuds les opérateurs $\{+, ., *\}$.

Intérêt .

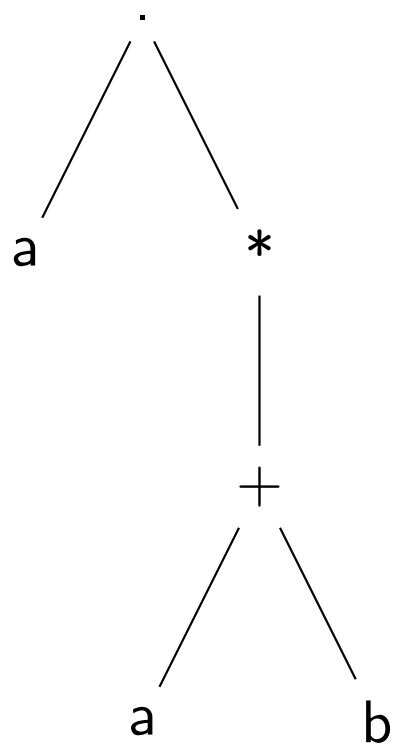
Par exemple l'arbre associé à l'expression rationnelle $a(a+b)^*$ est

Arbre binaire associé à une expression rationnelle

On peut naturellement associer à toute expression rationnelle un arbre binaire où les feuilles sont éléments de $\{\emptyset, \varepsilon\} \cup \Sigma$ et les nœuds les opérateurs $\{+, ., *\}$.

Intérêt .

Par exemple l'arbre associé à l'expression rationnelle $a(a+b)^*$ est



Expressions équivalentes

Expressions équivalentes

La correspondance entre expressions et langages n'est pas bijective

Expressions équivalentes

La correspondance entre expressions et langages n'est pas bijective

Définition

Deux expressions rationnelles sont dites équivalentes si elles dénotent le même langage.

Expressions équivalentes

La correspondance entre expressions et langages n'est pas bijective

Définition

Deux expressions rationnelles sont dites équivalentes si elles dénotent le même langage.

Remarques : – problèmes difficiles.

Expressions équivalentes

La correspondance entre expressions et langages n'est pas bijective

Définition

Deux expressions rationnelles sont dites équivalentes si elles dénotent le même langage.

Remarques : – problèmes difficiles.

– Deux simplifications relatives à $\emptyset$ et à ε.

Expressions équivalentes

La correspondance entre expressions et langages n'est pas bijective

Définition

Deux expressions rationnelles sont dites équivalentes si elles dénotent le même langage.

Remarques : – problèmes difficiles.
– Deux simplifications relatives à $\emptyset$ et à ε.

Proposition

Si le langage dénoté par l'expression rationnelle e est non vide, il existe une expression rationnelle e' équivalente à e ne contenant pas le symbole $\emptyset$.

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on peut poser $e' = e$.

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on peut poser $e' = e$.
- Si $e = e_1 + e_2$, alors $L = L_1 \cup L_2$, où L_i est le langage dénoté par e_i. Trois cas sont alors possibles

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on peut poser $e' = e$.
- Si $e = e_1 + e_2$, alors $L = L_1 \cup L_2$, où L_i est le langage dénoté par e_i. Trois cas sont alors possibles
  - Si $L_1 = \emptyset$, $L = L_2 \neq \emptyset$ et par hypothèse d'induction, il existe une expression régulière e_2' sans symbole $\emptyset$ dénotant $L_2 = L$ donc telle que $e_2' \equiv e$. Cas où $L_2 = \emptyset$ analogue

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on peut poser $e' = e$.
- Si $e = e_1 + e_2$, alors $L = L_1 \cup L_2$, où L_i est le langage dénoté par e_i. Trois cas sont alors possibles
  - Si $L_1 = \emptyset$, $L = L_2 \neq \emptyset$ et par hypothèse d'induction, il existe une expression régulière e'_2 sans symbole $\emptyset$ dénotant $L_2 = L$ donc telle que $e'_2 \equiv e$. Cas où $L_2 = \emptyset$ analogue
  - Sinon, L_1 et L_2 sont non vides donc, par hypothèse d'induction, il existe deux expressions rationnelles e'_1 et e'_2 ne comportant pas le symbole $\emptyset$ telles que $e'_1 \equiv e_1$ et $e'_2 \equiv e_2$ et alors $e' = e'_1 + e'_2$ est une expression sans $\emptyset$ équivalente à e.

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on peut poser $e' = e$.
- Si $e = e_1 + e_2$, alors $L = L_1 \cup L_2$, où L_i est le langage dénoté par e_i. Trois cas sont alors possibles
  - Si $L_1 = \emptyset$, $L = L_2 \neq \emptyset$ et par hypothèse d'induction, il existe une expression régulière e'_2 sans symbole $\emptyset$ dénotant $L_2 = L$ donc telle que $e'_2 \equiv e$. Cas où $L_2 = \emptyset$ analogue
  - Sinon, L_1 et L_2 sont non vides donc, par hypothèse d'induction, il existe deux expressions rationnelles e'_1 et e'_2 ne comportant pas le symbole $\emptyset$ telles que $e'_1 \equiv e_1$ et $e'_2 \equiv e_2$ et alors $e' = e'_1 + e'_2$ est une expression sans $\emptyset$ équivalente à e.
- Si $e = e_1 e_2$ aucune des deux expressions rationnelles e_1 et e_2 ne peut dénoter le langage vide (car la concaténation de deux langages dont l'un est l'ensemble vide est l'ensemble vide), donc par hypothèse d'induction, il existe $e'_1 \equiv e_1$ et $e'_2 \equiv e_2$ telles que e'_1 et e'_2 ne comportent pas le symbole $\emptyset$ et alors $e' = e'_1 e'_2$ est une expression équivalente à e ne comportant pas le symbole $\emptyset$.

Dém : $L \neq \emptyset$ dénoté par e. Raisonnt par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on peut poser $e' = e$.
- Si $e = e_1 + e_2$, alors $L = L_1 \cup L_2$, où L_i est le langage dénoté par e_i. Trois cas sont alors possibles
  - Si $L_1 = \emptyset$, $L = L_2 \neq \emptyset$ et par hypothèse d'induction, il existe une expression régulière e'_2 sans symbole $\emptyset$ dénotant $L_2 = L$ donc telle que $e'_2 \equiv e$. Cas où $L_2 = \emptyset$ analogue
  - Sinon, L_1 et L_2 sont non vides donc, par hypothèse d'induction, il existe deux expressions rationnelles e'_1 et e'_2 ne comportant pas le symbole $\emptyset$ telles que $e'_1 \equiv e_1$ et $e'_2 \equiv e_2$ et alors $e' = e'_1 + e'_2$ est une expression sans $\emptyset$ équivalente à e.
- Si $e = e_1 e_2$ aucune des deux expressions rationnelles e_1 et e_2 ne peut dénoter le langage vide (car la concaténation de deux langages dont l'un est l'ensemble vide est l'ensemble vide), donc par hypothèse d'induction, il existe $e'_1 \equiv e_1$ et $e'_2 \equiv e_2$ telles que e'_1 et e'_2 ne comportent pas le symbole $\emptyset$ et alors $e' = e'_1 e'_2$ est une expression équivalente à e ne comportant pas le symbole $\emptyset$.

- Si $e = e_1^*$ alors ou bien e_1 dénote le langage vide auquel cas $e \equiv \varepsilon$ ou bien par hypothèse d'induction, il existe une expression e_1' ne comportant pas le symbole $\emptyset$ telle que $e_1' \equiv e_1$ et alors $e \equiv (e_1')^*$ expression rationnelle ne comportant pas le symbole $\emptyset$ ♯

- Si $e = e_1^*$ alors ou bien e_1 dénote le langage vide auquel cas $e \equiv \varepsilon$ ou bien par hypothèse d'induction, il existe une expression e_1' ne comportant pas le symbole $\emptyset$ telle que $e_1' \equiv e_1$ et alors $e \equiv (e_1')^*$ expression rationnelle ne comportant pas le symbole $\emptyset$ ♯

Proposition

Si le langage dénoté par l'expression rationnelle e est non vide, il existe une expression rationnelle e' ne contenant ni le symbole $\emptyset$, ni le symbole ε telle que e soit équivalente à ε, e' ou $e' + \varepsilon$.

- Si $e = e_1^*$ alors ou bien e_1 dénote le langage vide auquel cas $e \equiv \varepsilon$ ou bien par hypothèse d'induction, il existe une expression e_1' ne comportant pas le symbole $\emptyset$ telle que $e_1' \equiv e_1$ et alors $e \equiv (e_1')^*$ expression rationnelle ne comportant pas le symbole $\emptyset$ ♯

Proposition

Si le langage dénoté par l'expression rationnelle e est non vide, il existe une expression rationnelle e' ne contenant ni le symbole $\emptyset$, ni le symbole ε telle que e soit équivalente à ε, e' ou $e' + \varepsilon$.

Démonstration : D'après la proposition précédente, on peut supposer que e ne comporte pas le symbole $\emptyset$. On raisonne par induction sur e.

- Si $e = e_1^*$ alors ou bien e_1 dénote le langage vide auquel cas $e \equiv \varepsilon$ ou bien par hypothèse d'induction, il existe une expression e_1' ne comportant pas le symbole $\emptyset$ telle que $e_1' \equiv e_1$ et alors $e \equiv (e_1')^*$ expression rationnelle ne comportant pas le symbole $\emptyset$ ♯

Proposition

Si le langage dénoté par l'expression rationnelle e est non vide, il existe une expression rationnelle e' ne contenant ni le symbole $\emptyset$, ni le symbole ε telle que e soit équivalente à ε, e' ou $e' + \varepsilon$.

Démonstration : D'après la proposition précédente, on peut supposer que e ne comporte pas le symbole $\emptyset$. On raisonne par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on a le résultat souhaité.

- Si $e = e_1^*$ alors ou bien e_1 dénote le langage vide auquel cas $e \equiv \varepsilon$ ou bien par hypothèse d'induction, il existe une expression e_1' ne comportant pas le symbole $\emptyset$ telle que $e_1' \equiv e_1$ et alors $e \equiv (e_1')^*$ expression rationnelle ne comportant pas le symbole $\emptyset$ ♯

Proposition

Si le langage dénoté par l'expression rationnelle e est non vide, il existe une expression rationnelle e' ne contenant ni le symbole $\emptyset$, ni le symbole ε telle que e soit équivalente à ε, e' ou $e' + \varepsilon$.

Démonstration : D'après la proposition précédente, on peut supposer que e ne comporte pas le symbole $\emptyset$. On raisonne par induction sur e.

- Si $e = \varepsilon$ ou $e \in \Sigma$, on a le résultat souhaité.
- Si $e = e_1^*$ et qu'on note e_1' une expression sans $\emptyset$ ni ε telle que e_1 soit équivalente à ε, e_1' ou $\varepsilon + e_1'$ on a selon les cas e qui est équivalente à qe, ${e_1'}^*$ ou à $\varepsilon + {e_1'}^*$

- Si $e = e_1 + e_2$, on applique l'hypothèse d'induction à e_1 et e_2 : il existe e_1' et e_2' expressions régulières sans $\emptyset$ ni ε telles que e_i soit équivalente à ε, e_i' ou $e_1' + \varepsilon$.
  Cela nous donne neuf cas possibles

- Si $e = e_1 + e_2$, on applique l'hypothèse d'induction à e_1 et e_2 : il existe e_1' et e_2' expressions régulières sans $\emptyset$ ni ε telles que e_i soit équivalente à ε, e_i' ou $e_1' + \varepsilon$.
  Cela nous donne neuf cas possibles

$+$	ε	e_2'	$\varepsilon + e_2'$
ε	ε	$\varepsilon + e_2'$	$\varepsilon + e_2'$
e_1'	$\varepsilon + e_1'$	$e_1' + e_2'$	$\varepsilon + (e_1' + e_2')$
$\varepsilon + e_1'$	$\varepsilon + e_1'$	$\varepsilon + (e_1' + e_2')$	$\varepsilon + (e_1' + e_2')$

- Si $e = e_1 + e_2$, on applique l'hypothèse d'induction à e_1 et e_2 : il existe e_1' et e_2' expressions régulières sans $\emptyset$ ni ε telles que e_i soit équivalente à ε, e_i' ou $e_1' + \varepsilon$.
  Cela nous donne neuf cas possibles

$+$	ε	e_2'	$\varepsilon + e_2'$
ε	ε	$\varepsilon + e_2'$	$\varepsilon + e_2'$
e_1'	$\varepsilon + e_1'$	$e_1' + e_2'$	$\varepsilon + (e_1' + e_2')$
$\varepsilon + e_1'$	$\varepsilon + e_1'$	$\varepsilon + (e_1' + e_2')$	$\varepsilon + (e_1' + e_2')$

- Si $e = e_1 e_2$, on a de même :

- Si $e = e_1 + e_2$, on applique l'hypothèse d'induction à e_1 et e_2 : il existe e_1' et e_2' expressions régulières sans $\emptyset$ ni ε telles que e_i soit équivalente à ε, e_i' ou $e_1' + \varepsilon$.
  Cela nous donne neuf cas possibles

$+$	ε	e_2'	$\varepsilon + e_2'$
ε	ε	$\varepsilon + e_2'$	$\varepsilon + e_2'$
e_1'	$\varepsilon + e_1'$	$e_1' + e_2'$	$\varepsilon + (e_1' + e_2')$
$\varepsilon + e_1'$	$\varepsilon + e_1'$	$\varepsilon + (e_1' + e_2')$	$\varepsilon + (e_1' + e_2')$

- Si $e = e_1 e_2$, on a de même :

$\cdot$	ε	e_2'	$\varepsilon + e_2'$
ε	ε	e_2'	$\varepsilon + e_2'$
e_1'	e_1'	$e_1' e_2'$	$e_1' + e_1' e_2'$
$\varepsilon + e_1'$	$\varepsilon + e_1'$	$e_2' + e_1' e_2'$	$\varepsilon + (e_1' + e_2' + e_1' e_2')$

♯

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

```
type regexp =
  | Epsilon
```

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

```
type regexp =
  | Epsilon
  | Const of string
```

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

```
type regexp =
  | Epsilon
  | Const of string
  | Somme of regexp * regexp
```

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

```
type regexp =
  | Epsilon
  | Const of string
  | Somme of regexp * regexp
  | Concat of regexp * regexp
```

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

```
type regexp =
  | Epsilon
  | Const of string
  | Somme of regexp * regexp
  | Concat of regexp * regexp
  | Etoile of regexp ;;
```

Par exemple l'expression rationnelle $a(a+b)^*$ sera représentée avec ce type par :

```
let e
=
```

Implémentation en OCaml des expressions rationnelles

On peut représenter en machine les expressions régulières sans symbole $\emptyset$ avec le type somme suivant :

```
type regexp =
  | Epsilon
  | Const of string
  | Somme of regexp * regexp
  | Concat of regexp * regexp
  | Etoile of regexp ;;
```

Par exemple l'expression rationnelle $a(a+b)^*$ sera représentée avec ce type par :

```
let e
=Concat(Const("a"),Etoile(Somme(Const("a"),Const("b"))));;
```

Définition

Un langage L sur l'alphabet Σ est dit local s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $B \subset \Sigma^2$ tels que
pour tout mot $m \in \Sigma^* \setminus \{\varepsilon\}$ s'écrivant $m = u_1 u_2 \dots u_n$, on a l'équivalence :

$$m \in L \iff \left(u_1 \in I, u_n \in F, \text{ et } \forall\, i \in [\![1, n-1]\!],\ u_i u_{i+1} \in B\right)$$

Définition

Un langage L sur l'alphabet Σ est dit local s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $B \subset \Sigma^2$ tels que
pour tout mot $m \in \Sigma^* \setminus \{\varepsilon\}$ s'écrivant $m = u_1 u_2 \dots u_n$, on a l'équivalence :

$$m \in L \iff \left(u_1 \in I, u_n \in F, \text{ et } \forall\, i \in [\![1, n-1]\!],\ u_i u_{i+1} \in B\right)$$

Remarque : À propos du mot vide.

Définition

Un langage L sur l'alphabet Σ est dit local s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $B \subset \Sigma^2$ tels que
pour tout mot $m \in \Sigma^* \setminus \{\varepsilon\}$ s'écrivant $m = u_1 u_2 \dots u_n$, on a l'équivalence :

$$m \in L \iff (u_1 \in I, u_n \in F, \text{ et } \forall\, i \in [\![1, n-1]\!],\ u_i u_{i+1} \in B)$$

Remarque : À propos du mot vide.
Notation : Si L est un langage local décrit par les ensembles I, F et B de définition précédente, on notera

$$L = L_{I,B,F,bo}$$

où bo est le booléen égal à true si ε appartient à L et à false sinon.

Définition

Un langage L sur l'alphabet Σ est dit local s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $B \subset \Sigma^2$ tels que
pour tout mot $m \in \Sigma^* \setminus \{\varepsilon\}$ s'écrivant $m = u_1 u_2 \dots u_n$, on a l'équivalence :

$$m \in L \iff (u_1 \in I, u_n \in F, \text{ et } \forall\, i \in [\![1, n-1]\!],\ u_i u_{i+1} \in B)$$

Remarque : À propos du mot vide.
Notation : Si L est un langage local décrit par les ensembles I, F et B de définition précédente, on notera

$$L = L_{I,B,F,bo}$$

où bo est le booléen égal à true si ε appartient à L et à false sinon.

Exemples : – $\emptyset$ et $\{\varepsilon\}$

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,
– $L = \{a\}$

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,
– $L = \{a\}$ est local, défini par $I = F = \{a\}$, $B = \emptyset$ et $\varepsilon \notin L$,

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,
– $L = \{a\}$ est local, défini par $I = F = \{a\}$, $B = \emptyset$ et $\varepsilon \notin L$,
– $L = \{(ab)^n\,;\, n \in \mathbb{N}\}$

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,
– $L = \{a\}$ est local, défini par $I = F = \{a\}$, $B = \emptyset$ et $\varepsilon \notin L$,
– $L = \{(ab)^n\,;\, n \in \mathbb{N}\}$ est local, défini par $I = \{a\}$, $B = \{ab, ba\}$, $F = \{b\}$ et $\varepsilon \in L$,

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,
– $L = \{a\}$ est local, défini par $I = F = \{a\}$, $B = \emptyset$ et $\varepsilon \notin L$,
– $L = \{(ab)^n\,;\, n \in \mathbb{N}\}$ est local, défini par $I = \{a\}$, $B = \{ab, ba\}$, $F = \{b\}$ et $\varepsilon \in L$,
– $L' = \{(ab)^n\,;\, n \in \mathbb{N}^*\}$

Exemples : – $\emptyset$ et $\{\varepsilon\}$ sont des langages locaux, définis par $I = B = F = \emptyset$,
– $L = \{a\}$ est local, défini par $I = F = \{a\}$, $B = \emptyset$ et $\varepsilon \notin L$,
– $L = \{(ab)^n\,;\, n \in \mathbb{N}\}$ est local, défini par $I = \{a\}$, $B = \{ab, ba\}$, $F = \{b\}$ et $\varepsilon \in L$,
– $L' = \{(ab)^n\,;\, n \in \mathbb{N}^*\}$ est également local, défini par les mêmes ensembles I, B, F mais avec cette fois $\varepsilon \notin L'$.

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F
– l'ensemble des facteurs de taille 2 des mots de L est inclus dans B

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F
– l'ensemble des facteurs de taille 2 des mots de L est inclus dans B
mais ces inclusions ne sont pas forcément des égalités car, par exemple si $\Sigma = \{a, b\}$, et $L = \{a^n\ ;\ n \in \mathbb{N}\}$, L est local, défini par $I = F = \{a\}$, $B = \{aa\}$ et $\varepsilon \in L$

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F
– l'ensemble des facteurs de taille 2 des mots de L est inclus dans B
mais ces inclusions ne sont pas forcément des égalités car, par exemple si $\Sigma = \{a, b\}$, et $L = \{a^n\ ;\ n \in \mathbb{N}\}$, L est local, défini par $I = F = \{a\}$, $B = \{aa\}$ et $\varepsilon \in L$ mais il est également défini par $I = \{a, b\}$, $B = \{aa, ab\}$, $F = \{a\}$ et $\varepsilon \in L$.

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F
– l'ensemble des facteurs de taille 2 des mots de L est inclus dans B
mais ces inclusions ne sont pas forcément des égalités car, par exemple si $\Sigma = \{a, b\}$, et $L = \{a^n\ ;\ n \in \mathbb{N}\}$, L est local, défini par $I = F = \{a\}$, $B = \{aa\}$ et $\varepsilon \in L$ mais il est également défini par $I = \{a, b\}$, $B = \{aa, ab\}$, $F = \{a\}$ et $\varepsilon \in L$.

Exercice : Dire si les langages suivants sont locaux.

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F
– l'ensemble des facteurs de taille 2 des mots de L est inclus dans B
mais ces inclusions ne sont pas forcément des égalités car, par exemple si $\Sigma = \{a, b\}$, et $L = \{a^n\ ;\ n \in \mathbb{N}\}$, L est local, défini par $I = F = \{a\}$, $B = \{aa\}$ et $\varepsilon \in L$ mais il est également défini par $I = \{a, b\}$, $B = \{aa, ab\}$, $F = \{a\}$ et $\varepsilon \in L$.

Exercice : Dire si les langages suivants sont locaux.
– $L = \{a^n b^p\ ;\ (n, p) \in \mathbb{N}^2\}$

Remarque : Si $L = L_{I,B,F,b}$ est local on peut affirmer que :
– l'ensemble des préfixes de taille 1 des mots de L est inclus dans I
– l'ensemble des suffixes de taille 1 des mots de L est inclus dans F
– l'ensemble des facteurs de taille 2 des mots de L est inclus dans B
mais ces inclusions ne sont pas forcément des égalités car, par exemple si $\Sigma = \{a, b\}$, et $L = \{a^n\ ;\ n \in \mathbb{N}\}$, L est local, défini par $I = F = \{a\}$, $B = \{aa\}$ et $\varepsilon \in L$ mais il est également défini par $I = \{a, b\}$, $B = \{aa, ab\}$, $F = \{a\}$ et $\varepsilon \in L$.

Exercice : Dire si les langages suivants sont locaux.
– $L = \{a^n b^p\ ;\ (n, p) \in \mathbb{N}^2\}$
– $L' = \{a^n b^n\ ;\ n \in \mathbb{N}\}$

Remarque : En introduisant $N = \Sigma^2 \setminus B$, on peut donner une définition plus synthétique des langages locaux

Remarque : En introduisant $N = \Sigma^2 \setminus B$, on peut donner une définition plus synthétique des langages locaux

Proposition

Un langage L sur l'alphabet Σ est local si et seulement s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $N \subset \Sigma^2$ tels que :

$$L \setminus \{\varepsilon\} = (I.\Sigma^* \cap \Sigma^*.F) \setminus (\Sigma^*.N.\Sigma^*)$$

Remarque : En introduisant $N = \Sigma^2 \setminus B$, on peut donner une définition plus synthétique des langages locaux

Proposition

Un langage L sur l'alphabet Σ est local si et seulement s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $N \subset \Sigma^2$ tels que :

$$L \setminus \{\varepsilon\} = (I.\Sigma^* \cap \Sigma^*.F) \setminus (\Sigma^*.N.\Sigma^*)$$

Démonstration :

Remarque : En introduisant $N = \Sigma^2 \setminus B$, on peut donner une définition plus synthétique des langages locaux

Proposition

Un langage L sur l'alphabet Σ est local si et seulement s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $N \subset \Sigma^2$ tels que :

$$L \setminus \{\varepsilon\} = (I.\Sigma^* \cap \Sigma^*.F) \setminus (\Sigma^*.N.\Sigma^*)$$

Démonstration :

Remarque : Cette caractérisation montre qu'un langage local peut être construit à partir des trois langages rationnels $I.\Sigma^*$, $\Sigma^*.F$ et $\Sigma^*.N.\Sigma^*$ mais en utilisant des opérations non rationnelles : l'intersection et la différence.

Remarque : En introduisant $N = \Sigma^2 \setminus B$, on peut donner une définition plus synthétique des langages locaux

Proposition

Un langage L sur l'alphabet Σ est local si et seulement s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $N \subset \Sigma^2$ tels que :

$$L \setminus \{\varepsilon\} = (I.\Sigma^* \cap \Sigma^*.F) \setminus (\Sigma^*.N.\Sigma^*)$$

Démonstration :

Remarque : Cette caractérisation montre qu'un langage local peut être construit à partir des trois langages rationnels $I.\Sigma^*$, $\Sigma^*.F$ et $\Sigma^*.N.\Sigma^*$ mais en utilisant des opérations non rationnelles : l'intersection et la différence.

La réciproque n'est pas vraie comme le prouve le langage $L = \{ab\} \cup \{a^n\,;\, n \in \mathbb{N}\}$

Remarque : En introduisant $N = \Sigma^2 \setminus B$, on peut donner une définition plus synthétique des langages locaux

Proposition

Un langage L sur l'alphabet Σ est local si et seulement s'il existe $I \subset \Sigma$, $F \subset \Sigma$ et $N \subset \Sigma^2$ tels que :

$$L \setminus \{\varepsilon\} = (I.\Sigma^* \cap \Sigma^*.F) \setminus (\Sigma^*.N.\Sigma^*)$$

Démonstration :

Remarque : Cette caractérisation montre qu'un langage local peut être construit à partir des trois langages rationnels $I.\Sigma^*$, $\Sigma^*.F$ et $\Sigma^*.N.\Sigma^*$ mais en utilisant des opérations non rationnelles : l'intersection et la différence.

La réciproque n'est pas vraie comme le prouve le langage $L = \{ab\} \cup \{a^n\ ;\ n \in \mathbb{N}\}$ qui est rationnel mais n'est pas local.

Opérations sur les langages locaux

Opérations sur les langages locaux

Proposition

L'intersection de deux langages locaux est un langage local

Démonstration :

Opérations sur les langages locaux

Proposition

L'intersection de deux langages locaux est un langage local

Démonstration :

Remarque : La réunion ou la concaténation de deux langages locaux n'est pas forcément un langage local :

Opérations sur les langages locaux

Proposition

L'intersection de deux langages locaux est un langage local

Démonstration :

Remarque : La réunion ou la concaténation de deux langages locaux n'est pas forcément un langage local : $L_1 = \{ab\}$ et $L_2 = \{a^n \, ; \, n \in \mathbb{N}\}$.

Proposition

Si L_1 et L_2 sont deux langages locaux définis sur des alphabets disjoints, alors $L_1 \cup L_2$ est encore un langage local

Opérations sur les langages locaux

Proposition

L'intersection de deux langages locaux est un langage local

Démonstration :

Remarque : La réunion ou la concaténation de deux langages locaux n'est pas forcément un langage local : $L_1 = \{ab\}$ et $L_2 = \{a^n \, ; \, n \in \mathbb{N}\}$.

Proposition

Si L_1 et L_2 sont deux langages locaux définis sur des alphabets disjoints, alors $L_1 \cup L_2$ est encore un langage local

Proposition

Si L_1 et L_2 sont deux langages locaux définis sur des alphabets disjoints, alors $L_1.L_2$ est encore un langage local

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$. Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.
Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.
Si $u \in L^* \setminus \{\varepsilon\}$, u s'écrit $u_1 u_2 \dots u_k$ avec tous les u_i dans L.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.
Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.
Si $u \in L^* \setminus \{\varepsilon\}$, u s'écrit $u_1 u_2 \dots u_k$ avec tous les u_i dans L.
La première lettre de u est celle de u_1 et appartient donc à $I = I'$.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.
Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.
Si $u \in L^* \setminus \{\varepsilon\}$, u s'écrit $u_1 u_2 \dots u_k$ avec tous les u_i dans L.
La première lettre de u est celle de u_1 et appartient donc à $I = I'$.
La dernière lettre de u est celle de u_k et appartient à $F = F'$.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.
Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.
Si $u \in L^* \setminus \{\varepsilon\}$, u s'écrit $u_1 u_2 \dots u_k$ avec tous les u_i dans L.
La première lettre de u est celle de u_1 et appartient donc à $I = I'$.
La dernière lettre de u est celle de u_k et appartient à $F = F'$.
Les facteurs de longueur 2 de u sont soit des facteurs de l'un des u_i soit un mot de deux lettres constitué de la dernière lettre d'un u_i et de la première lettre de u_{i+1} donc un mot de $F.I \setminus B'$.
On a donc $u \in L_{I',B',F',b'}$. Par conséquent $L^* \subset L_{I',B',F',b'}$.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.
Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.
Si $u \in L^* \setminus \{\varepsilon\}$, u s'écrit $u_1 u_2 \dots u_k$ avec tous les u_i dans L.
La première lettre de u est celle de u_1 et appartient donc à $I = I'$.
La dernière lettre de u est celle de u_k et appartient à $F = F'$.
Les facteurs de longueur 2 de u sont soit des facteurs de l'un des u_i soit un mot de deux lettres constitué de la dernière lettre d'un u_i et de la première lettre de u_{i+1} donc un mot de $F.I \setminus B'$.
On a donc $u \in L_{I',B',F',b'}$. Par conséquent $L^* \subset L_{I',B',F',b'}$.
Réciproquement si $u \in L_{I',B',F',b'} \setminus \{\varepsilon\}$.

Proposition

Si L est un langage local, L^* est un langage local.

Démonstration : Par hypothèse L est de la forme $L = L_{I,B,F,bo}$.
Posons $I' = I$, $F' = F$, $B' = B \cup F.I$, $b' = true$ et montrons que $L^* = L_{I',B',F',b'}$.
Si $u \in L^* \setminus \{\varepsilon\}$, u s'écrit $u_1 u_2 \dots u_k$ avec tous les u_i dans L.
La première lettre de u est celle de u_1 et appartient donc à $I = I'$.
La dernière lettre de u est celle de u_k et appartient à $F = F'$.
Les facteurs de longueur 2 de u sont soit des facteurs de l'un des u_i soit un mot de deux lettres constitué de la dernière lettre d'un u_i et de la première lettre de u_{i+1} donc un mot de $F.I \setminus B'$.
On a donc $u \in L_{I',B',F',b'}$. Par conséquent $L^* \subset L_{I',B',F',b'}$.
Réciproquement si $u \in L_{I',B',F',b'} \setminus \{\varepsilon\}$.
Parmi les facteurs de longueur 2 de u considérons ceux appartenant à $F.I$: ils permettent de factoriser u en $u_1 u_2 \dots u_k$ avec pour tout i, $u_i \in L_{I,B,F,b} \setminus \{\varepsilon\} \subset L$ et donc $u \in L^*$. ♯

Expressions rationnelles linéaires

Expressions rationnelles linéaires

Une expression rationnelle générale ne dénote pas forcément un langage local

Expressions rationnelles linéaires

Une expression rationnelle générale ne dénote pas forcément un langage local

Définition

Une expression rationnelle e sur l'alphabet Σ est dite linéaire si tout caractère de Σ apparaît au plus une fois dans e.

Expressions rationnelles linéaires

Une expression rationnelle générale ne dénote pas forcément un langage local

Définition

Une expression rationnelle e sur l'alphabet Σ est dite linéaire si tout caractère de Σ apparaît au plus une fois dans e.

Exemples : sur $\Sigma = \{a, b, c\}$, $ab + a^*$ n'est pas linéaire mais $ab + c^*$ l'est.

Expressions rationnelles linéaires

Une expression rationnelle générale ne dénote pas forcément un langage local

Définition

Une expression rationnelle e sur l'alphabet Σ est dite linéaire si tout caractère de Σ apparaît au plus une fois dans e.

Exemples : sur $\Sigma = \{a, b, c\}$, $ab + a^*$ n'est pas linéaire mais $ab + c^*$ l'est.

Proposition

Tout langage dénoté par une expression rationnelle linéaire est local.

Expressions rationnelles linéaires

Une expression rationnelle générale ne dénote pas forcément un langage local

Définition

Une expression rationnelle e sur l'alphabet Σ est dite linéaire si tout caractère de Σ apparaît au plus une fois dans e.

Exemples : sur $\Sigma = \{a, b, c\}$, $ab + a^*$ n'est pas linéaire mais $ab + c^*$ l'est.

Proposition

Tout langage dénoté par une expression rationnelle linéaire est local.

Démonstration : Par induction structurelle sur l'expression rationnelle linéaire e.

Remarque : La réciproque du résultat précédent est fausse : par exemple $L = \{a^n\,,\ n \in \mathbb{N}^*\}$

Algorithmes de détermination des ensembles I, B et F

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
  | Concat (e1,e2) -> motvide e1 && motvide e2
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
  | Concat (e1,e2) -> motvide e1 && motvide e2
  | Etoile _ -> true;;
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
  | Concat (e1,e2) -> motvide e1 && motvide e2
  | Etoile _ -> true;;

let rec union l1 l2 = match l1 with
  | [] -> l2
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
  | Concat (e1,e2) -> motvide e1 && motvide e2
  | Etoile _ -> true;;

let rec union l1 l2 = match l1 with
  | [] -> l2
  | t :: q when List.mem t l2 -> union q l2
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
  | Concat (e1,e2) -> motvide e1 && motvide e2
  | Etoile _ -> true;;
```

```
let rec union l1 l2 = match l1 with
  | [] -> l2
  | t :: q when List.mem t l2 -> union q l2
  | t :: q -> t :: (union q l2);;
```

Algorithmes de détermination des ensembles I, B et F

Écrivons tout d'abord une fonction déterminant si le mot vide appartient au langage dénoté par une expression régulière :

```
let rec motvide e = match e with
  | Epsilon -> true
  | Const _ -> false
  | Somme (e1,e2) -> motvide e1 || motvide e2
  | Concat (e1,e2) -> motvide e1 && motvide e2
  | Etoile _ -> true;;
```

```
let rec union l1 l2 = match l1 with
  | [] -> l2
  | t :: q when List.mem t l2 -> union q l2
  | t :: q -> t :: (union q l2);;
```

```
let rec i e = match e with
```

```
let rec i e = match e with
  | Epsilon -> []
```

```
let rec i e = match e with
  | Epsilon -> []
  | Const a -> [a]
```

```
let rec i e = match e with
  | Epsilon -> []
  | Const a -> [a]
  | Somme (e1, e2) -> union (i e1) (i e2)
```

```
let rec i e = match e with
  | Epsilon -> []
  | Const a -> [a]
  | Somme (e1, e2) -> union (i e1) (i e2)
  | Concat(e1,e2) when motvide e1->union(i e1)(i e2)
```

```
let rec i e = match e with
  | Epsilon -> []
  | Const a -> [a]
  | Somme (e1, e2) -> union (i e1) (i e2)
  | Concat(e1,e2) when motvide e1->union(i e1)(i e2)
  | Concat (e1,e2) -> i e1
```

```
let rec i e = match e with
  | Epsilon -> []
  | Const a -> [a]
  | Somme (e1, e2) -> union (i e1) (i e2)
  | Concat(e1,e2) when motvide e1->union(i e1)(i e2)
  | Concat (e1,e2) -> i e1
  | Etoile e1 -> i e1;;
```

```
let rec i e = match e with
  | Epsilon -> []
  | Const a -> [a]
  | Somme (e1, e2) -> union (i e1) (i e2)
  | Concat(e1,e2) when motvide e1->union(i e1)(i e2)
  | Concat (e1,e2) -> i e1
  | Etoile e1 -> i e1;;
```

```
let rec f e = match e with
  | Epsilon -> []
  | Const a -> [a]
  | Somme (e1, e2) -> union (f e1) (f e2)
  | Concat(e1,e2) when motvide e2->union(f e1)(f e2)
  | Concat (e1,e2) -> f e2
  | Etoile e1 -> f e1;;
```

```
let rec produit l1 l2 = match (l1,l2) with
  | [] , _ -> []
  | _ , [] -> []
```

```
let rec produit l1 l2 = match (l1,l2) with
 | [] , _ -> []
 | _ , [] -> []
 | t1::q1, _ -> let l = List.map (fun x->t1^x) l2
                in union l (produit q1 l2);;
```

```
let rec produit l1 l2 = match (l1,l2) with
 | [] , _ -> []
 | _ , [] -> []
 | t1::q1, _ -> let l = List.map (fun x->t1^x) l2
                in union l (produit q1 l2);;
```

```
let rec b e = match e with
 | Epsilon -> []
 | Const a -> []
```

```
let rec produit l1 l2 = match (l1,l2) with
 | [] , _ -> []
 | _ , [] -> []
 | t1::q1, _ -> let l = List.map (fun x->t1^x) l2
                in union l (produit q1 l2);;
```

```
let rec b e = match e with
 | Epsilon -> []
 | Const a -> []
 | Somme (e1,e2) -> union (b e1) (b e2)
```

```
let rec produit l1 l2 = match (l1,l2) with
 | [] , _ -> []
 | _ , [] -> []
 | t1::q1, _ -> let l = List.map (fun x->t1^x) l2
                in union l (produit q1 l2);;
```

```
let rec b e = match e with
 | Epsilon -> []
 | Const a -> []
 | Somme (e1,e2) -> union (b e1) (b e2)
 | Concat (e1,e2) -> let l = union (b e1) (b e2) in
                union l (produit (f e1) (i e2))
```

```
let rec produit l1 l2 = match (l1,l2) with
 | [] , _ -> []
 | _ , [] -> []
 | t1::q1, _ -> let l = List.map (fun x->t1^x) l2
                in union l (produit q1 l2);;
```

```
let rec b e = match e with
 | Epsilon -> []
 | Const a -> []
 | Somme (e1,e2) -> union (b e1) (b e2)
 | Concat (e1,e2) -> let l = union (b e1) (b e2) in
                union l (produit (f e1) (i e2))
 | Etoile e1 -> union (b e1)(produit(f e1)(i e1));;
```